

**IBM Hack Challenge 2023**

**Project Title:**

**Identifying Patterns and Trends in Campus Placement Data using Machine Learning**

**Project Id:**

**SPS\_PRO\_3780**

**Submitted by:**

Team Name: Data Bytes

Vidul garg

Manav Bakliwal

**Abstract**

The project titled "Identifying Patterns and Trends in Campus Placement Data using Machine Learning" aims to leverage machine learning techniques to extract meaningful insights from campus placement data.

This data encompasses various aspects of students' academic journeys, such as their academic performance, skill sets, internship experiences, and eventual placement outcomes.

By discerning patterns and trends within this dataset, this project seeks to uncover factors that influence placement success. The ultimate goal is to facilitate the enhancement of the placement process through informed decision-making.

Employing tools like Python, Data Science, and Machine Learning, alongside web frameworks like Flask or Django, this project demonstrates how colleges and universities can harness the power of machine learning to gain actionable insights from placement data, thereby refining the placement process and offering targeted support to students for improved employability.

**Introduction**

In the realm of higher education, campus placements play a pivotal role in students' transition from academia to the professional world. The process entails intricate interactions among multiple variables, including academic performance, acquired skills, and internships, culminating in the desired outcome of successful placements.

The primary objective of this project is to provide colleges and universities with a deeper understanding of the factors that contribute to successful placements. This understanding is vital for enhancing the placement process and ensuring that students are well-prepared for the job market. Leveraging tools like Python, Data Science, and Machine Learning, we seek to offer insights that can guide institutions in offering targeted support to students, thereby improving their chances of employability.

As a testament to the project's potential, we draw inspiration from the diverse and comprehensive campus placement dataset available on platforms like Kaggle. By applying machine learning techniques to this dataset, we intend to equip colleges and universities with actionable knowledge that can enhance their placement processes, fostering an environment where students are better equipped for successful careers.

Through our exploration, we demonstrate how machine learning can transform campus placement data into a source of strategic advantage, paving the way for improved employability and enriched educational experiences.

**Project Formulation**

**Problem statement:**

Identifying Patterns And Trends In Campus Placement Data Using Machine Learning-

**Expected solution:**

By addressing the problem statement with machine learning techniques we can effectively identify patterns and trends in campus placement data considering the key considerations, colleges and universities can gain valuable insights from campus placement data using machine learning.

This allows them to identify factors affecting placement outcomes, improve the placement process, and provide targeted support to students to enhance their employability.

**Steps:**

1. Data collection from kaggle
2. Data Cleaning & Handling missing values
3. Data encoding
4. Data visualization
5. Data preprocessing (Feature scaling)
6. Model Training / Building
7. Model Deployment

**Model:**

**Data Collection:**

* Kaggle dataset: [https://www.kaggle.com/datasets/tejashvi14/engineering-placements-prediction](https://www.kaggle.com/datasets/benroshan/factors-affecting-campus-placement)

**Libraries required:**

import sklearn

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import MinMaxScaler

from sklearn.impute import SimpleImputer

import plotly

import plotly.graph\_objs as go

from plotly.subplots import make\_subplots

import matplotlib.pyplot as plt

from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear\_model import LogisticRegression

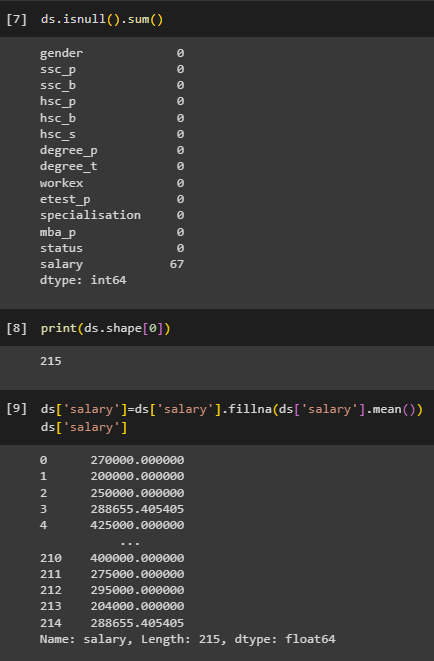
from sklearn.tree import DecisionTreeClassifier

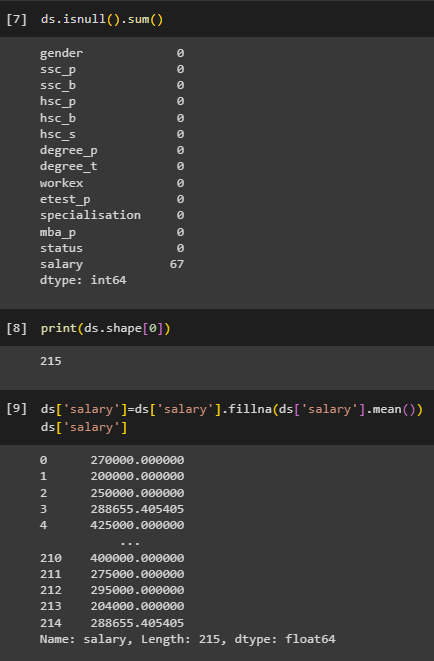
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report

import pickle

**Data Cleaning & Handling Missing Values:**

****

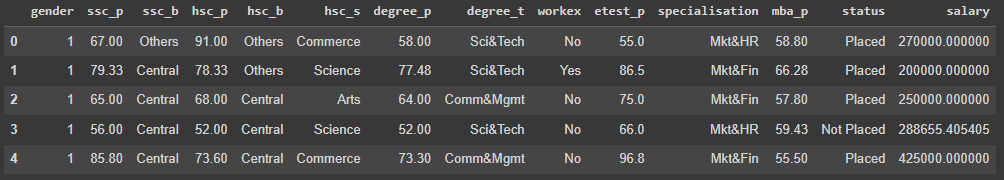
****

Data Encoding

# Manual Encoding

ds['gender']=ds['gender'].replace({'M':1,'F':0})

ds.head(5)



# Label Encoding

from sklearn.preprocessing import LabelEncoder

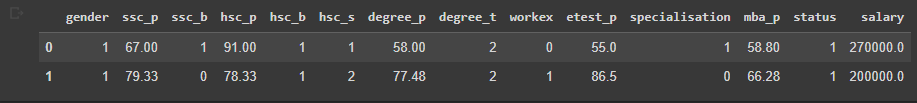
le=LabelEncoder()

l=['ssc\_b','hsc\_s','hsc\_b','degree\_t','workex','specialisation','status']

for i in l:

  ds[i]=le.fit\_transform(ds[i])

ds.head(2)



Data Visualization

Plot of every column :

for x in ds.columns[1:]:

    plt.hist(ds[ds["status"]=="Placed"][x], color='red',label='placed',alpha=0.7,density=True)

    plt.hist(ds[ds["status"]=="Not Placed"][x], color='pink',label='notplaced',alpha=0.7,density=True)

    plt.title(x)

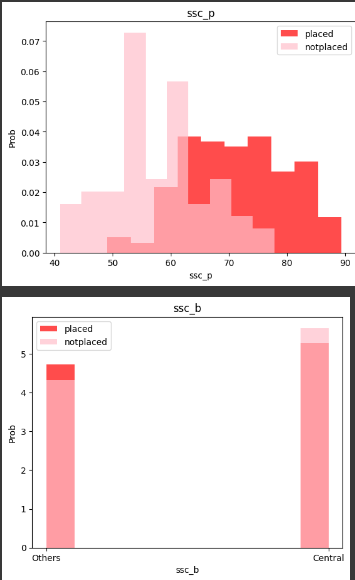
    plt.ylabel("Prob")

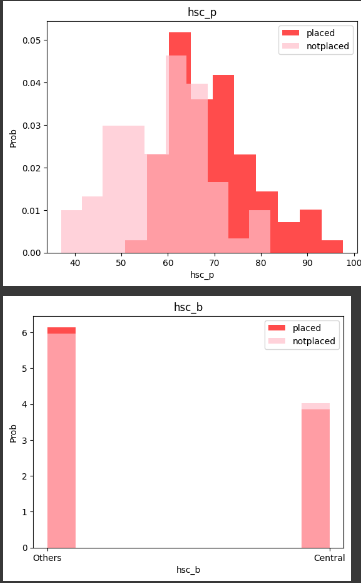
    plt.xlabel(x)

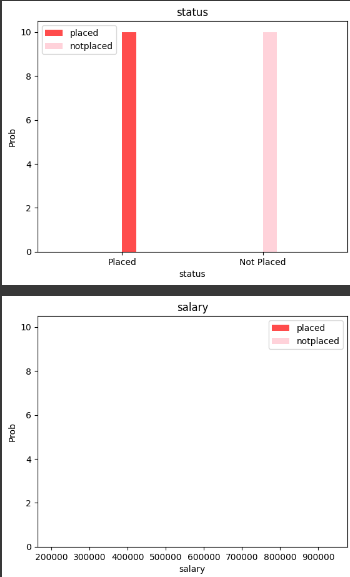
    plt.legend()

    plt.show()

    print()







From the above plot we got to know that salary column is not related to status whether the student is placed or not so we'll not include it in the feature for our model.

**HeatMap:**

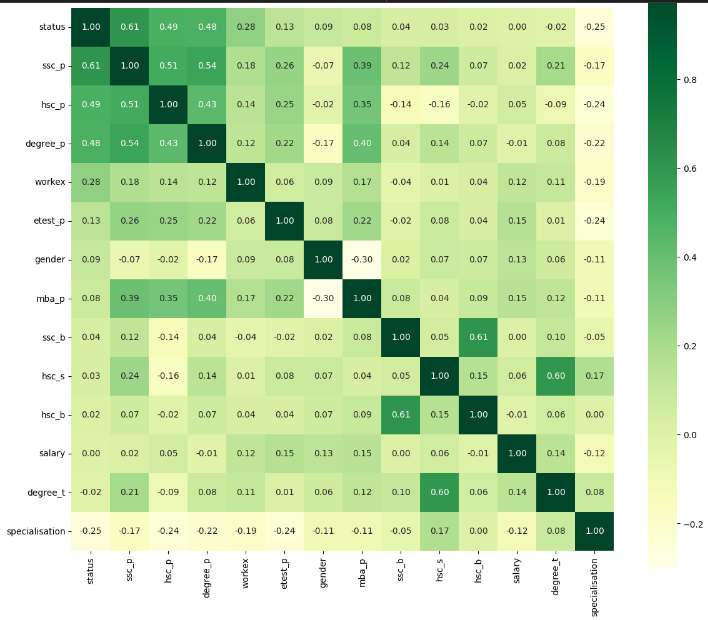
corrmat = ds.corr()

cols = corrmat.nlargest(k, 'status')['status'].index

cm = np.corrcoef(ds[cols].values.T)

plt.figure(figsize=(14,12))

hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot\_kws={'size': 10}, yticklabels=cols.values, xticklabels=cols.values, cmap='YlGn', linecolor='white')

****

**Q) HOW MANY STUDENTS PLACED ??**

print("Number of students placed: "+ str(len(ds[ds["status"]==1])))

print("Number of students not placed: "+ str(len(ds[ds["status"]==0]))+ "\n")

plt.bar([0],height=len(ds[ds["status"]==0]))    # [0] means x-coord=0

plt.bar([1],height=len(ds[ds["status"]==1]))    # [1] means x-coord=1

plt.xlabel("Status")

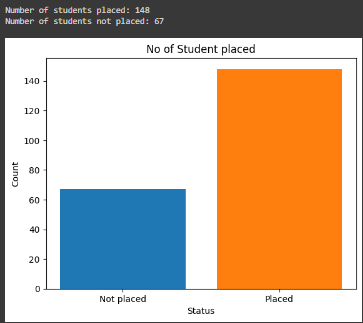
plt.ylabel("Count")

plt.xticks(np.arange(2), ('Not placed', 'Placed'))   # to label the bars

plt.title("No of Student placed")

plt.show()

plt.savefig("fig1.png")

****

**Q) Does percentage matters for one to get placed?**

#show the relation between diffrent qualification placement status usinng correlation.

# Calculate correlation between features and placement status

correlation\_matrix = ds.corr()

placement\_correlation = correlation\_matrix.loc['status', ['ssc\_p', 'hsc\_p', 'mba\_p', 'degree\_p', 'etest\_p']]

# Calculate the correlation percentages

correlation\_percentages = (placement\_correlation \* 100).round(1)

# Print the correlation percentages

for feature, percentage in correlation\_percentages.items():

    print(f"{feature} to placement : {percentage} %")

#Student Grades and Campus Placement

import plotly.offline as pyo

pyo.init\_notebook\_mode(connected=True)

# plotly.offline.init\_notebook\_mode(connected=True)

%matplotlib inline

trace1 = go.Bar(

    x = ['High School', 'Bachelor', 'MBA'],

    y = df\_grade[df\_grade['status']==0].drop('status', axis=1).values[0],

    name = 'Not Placed'

)

trace2 = go.Bar(

    x = ['High School', 'Bachelor', 'MBA'],

    y = df\_grade[df\_grade['status']==1].drop('status', axis=1).values[0],

    name = 'Placed'

)

data = [trace1, trace2]

layout = go.Layout(

    yaxis = dict(title = 'Grade'),

    xaxis = dict(title = 'Stage'),

    title = 'Student Grades and Campus Placement')

fig = go.Figure(data=data, layout=layout)

fig.show()

fig.write\_html("plot.html")

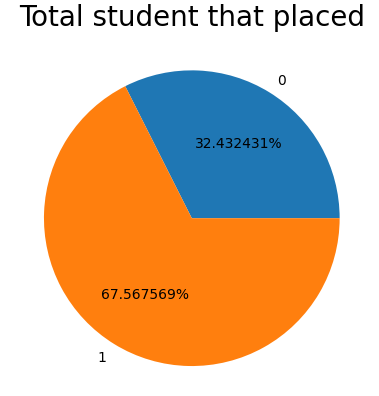
* From the Above graph we can say that percentage matters and ssc\_p feature data factor influenced a candidate in getting placed. when we see the correlation between features and placement then it shows that the ssc\_p data is more correlated to placement.
* We know that it is some tricky to say that senior secondary(ssc\_p)percentage is not so much help in real world placement. But After seeing the correlation between features and placement the we definitely say the percentage matters for getting placed.

**Pie Chart of ‘status’**

plt.pie(obj1.values,labels=obj1.index,autopct="%2f%%")

plt.title("Total student that placed",fontsize=20)

plt.savefig("fig4.png")



**Q) What is the percentage of female or male getting placement?**

def count\_genderwise\_status(gender=0,status=0):

    i=0

    for j in range(len(ds)-1):

        j=j+1

        if ds['gender'][j]==gender and ds['status'][j]==status:

            i+=1

    if gender==0 and status==0:

        print("Total female who don't get placement is: "+str(i))

    if gender==0 and status==1:

        print("Total female who get placement is: "+str(i))

    if gender==1 and status==0:

        print("Total male who don't get placement is: "+str(i))

    if gender==1 and status==1:

        print("Total male who get placement is: "+str(i))

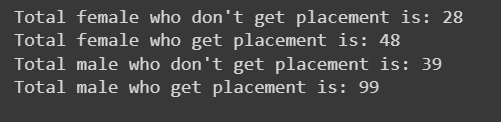
    return i

not\_placed\_female=count\_genderwise\_status(0,0)

placed\_female=count\_genderwise\_status(0,1)

not\_placed\_male=count\_genderwise\_status(1,0)

placed\_male=count\_genderwise\_status(1,1)

****

**Pie Chart of placement:**

#plot pie chart of placeement

plt.pie([22.33,46.51,31.17],

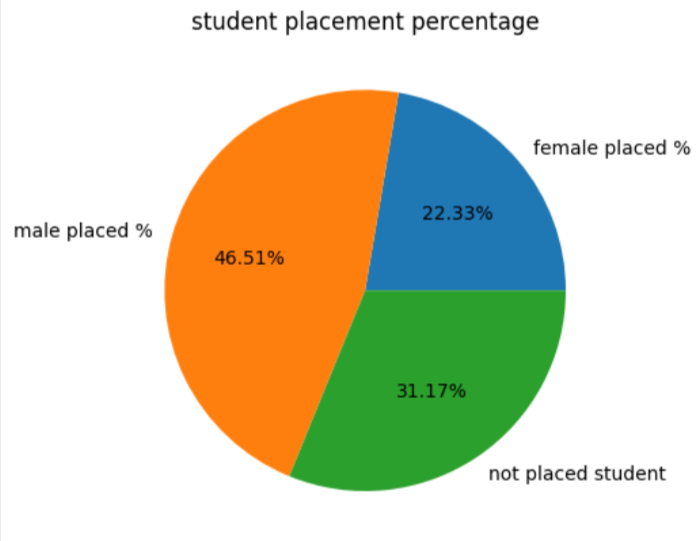
        labels=["female placed %","male placed %",'not placed student'],

        autopct='%1.2f%%')

plt.title('student placement percentage')

plt.show()

plt.savefig("fig5.png")

****

**Q) Which degree specialization is much demanded by corporate?**

# which specialisation is more demand in campus selection

plt.bar([1],height=len(ds[ds["specialisation"]==1]))

plt.bar([0],height=len(ds[ds["specialisation"]==0]))

plt.xlabel("specialisation in Mkt&Fin and Mkt&HR")

plt.ylabel("no.of specialisation")

print("specialisation in Mkt&HR "+ str(len(ds[ds["specialisation"]==1])))

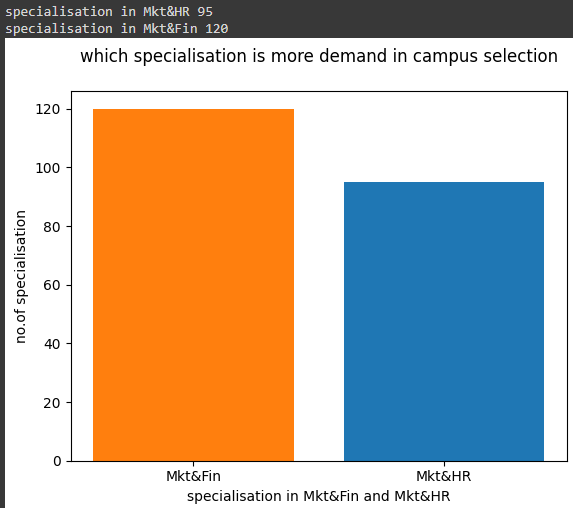
print("specialisation in Mkt&Fin "+ str(len(ds[ds["specialisation"]==0])))

plt.xticks(np.arange(2), ('Mkt&Fin', 'Mkt&HR'))

plt.title("which specialisation is more demand in campus selection\n")

plt.show()

plt.savefig("fig6.png")

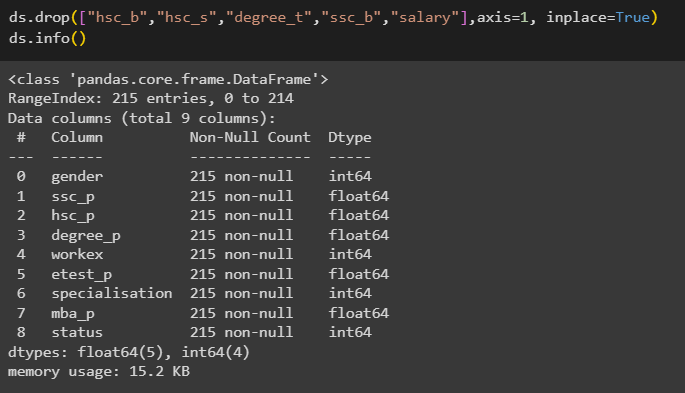
****

**Sns Pairplot:**



* We can say that stream and board of secondary and higher secondary doesn't matter in getting placement. Therefore, removing such columns:

Also from the plots, we got to know that salary doesn't matter for getting placed , So we'll remove salary column also.

****

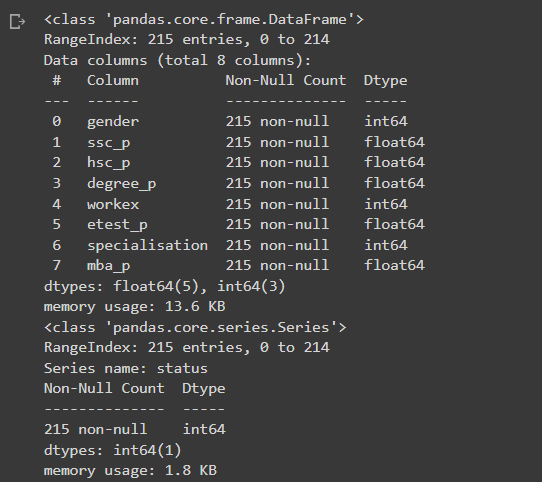
**Splitting Features and Labels:**

x=ds.iloc[:,:8]

y=ds.iloc[:,-1]

x.info()

y.info()



**Data Preprocessing (Feature Scaling):**

# Feature scaling (MinMax Scaler or Standard Scaler)

p=Pipeline([('min\_max',MinMaxScaler())

            ])

p

# Using MinMax Scaler as it's giving better results in the model after checking performance one by one

x=p.fit\_transform(x)

x=pd.DataFrame(x)

x.iloc[:5,:]

**TrainTestSplit:**

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import r2\_score

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.4,random\_state=32)

**Model Training:**

m1=KNeighborsClassifier()

m1.fit(x\_train,y\_train)

pred1=m1.predict(x\_test)

# r2\_score(y\_test,pred1)

print(classification\_report(y\_test,pred1))

m2=LogisticRegression()

m2.fit(x\_train,y\_train)

pred2=m2.predict(x\_test)

# r2\_score(y\_test,pred2)

print(classification\_report(y\_test,pred2))

m3=RandomForestClassifier(n\_estimators=100,random\_state=1)

m3.fit(x\_train,y\_train)

pred3=m3.predict(x\_test)

# r2\_score(y\_test,pred3)

print(classification\_report(y\_test,pred3))

m4=DecisionTreeClassifier()

m4.fit(x\_train,y\_train)

pred4=m4.predict(x\_test)

# r2\_score(y\_test,pred4)

print(classification\_report(y\_test,pred4))

**Converting to pickle file:**

# Taking 'm2' as our model and 'p' is our preprocessing pipeline

model\_with\_preprocessing = {'model': m2, 'preprocessing': p}

# Save the combined model and preprocessing

with open('combined\_model.pkl', 'wb') as f:

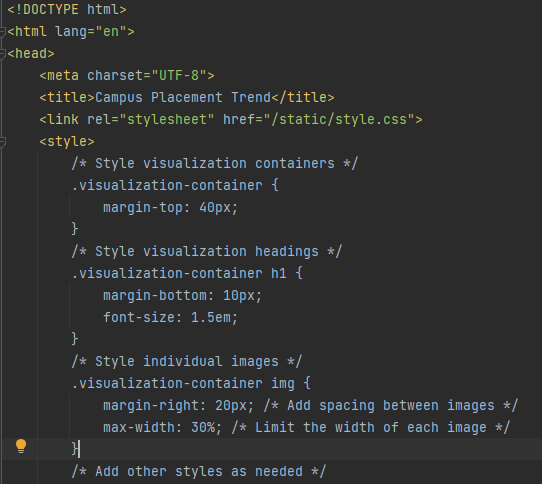
    pickle.dump(model\_with\_preprocessing, f)

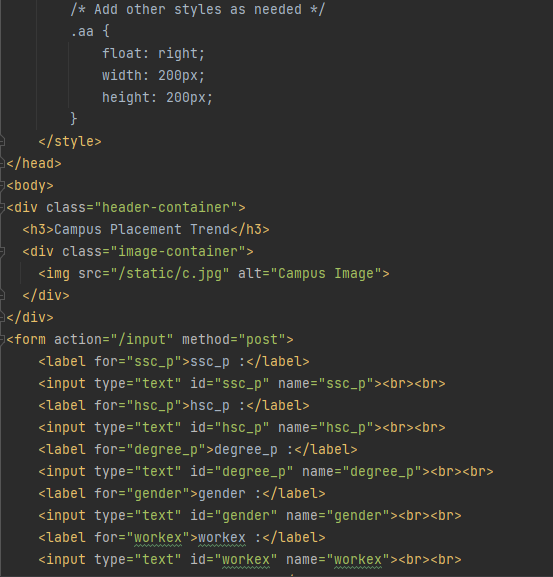
**Model Deployment Code:**

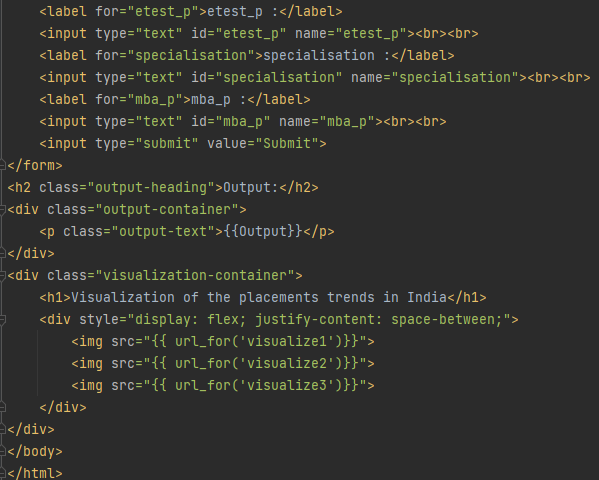
**Python Code:**

from flask import Flask, request, render\_template  
import pickle  
import logging  
  
app = Flask(\_\_name\_\_)  
  
# Set up logging  
logging.basicConfig(level=logging.DEBUG)  
  
try:  
 with open('combined\_model.pkl', 'rb') as f:  
 combined\_model = pickle.load(f)  
 model = combined\_model['model']  
 preprocessing = combined\_model['preprocessing']  
except Exception as e:  
 logging.error("Error loading the model: %s", str(e))  
  
@app.route('/')  
def home():  
 return render\_template('input.html')  
  
@app.route('/input', methods=['POST'])  
  
  
def pred():  
 try:  
 # Retrieve user input data  
 ssc\_p = float(request.form.get('ssc\_p'))  
 hsc\_p = float(request.form.get('hsc\_p'))  
 degree\_p = float(request.form.get('degree\_p'))  
 gender = int(request.form.get('gender'))  
 workex = int(request.form.get('workex'))  
 etest\_p = float(request.form.get('etest\_p'))  
 specialisation = int(request.form.get('specialisation'))  
 mba\_p = float(request.form.get('mba\_p'))  
  
 # Original input data  
 input\_data = [[ssc\_p, hsc\_p, degree\_p, gender, workex, etest\_p, specialisation, mba\_p]]  
  
 # Apply the same preprocessing as in Jupyter Notebook  
 scaled\_input\_data = preprocessing.fit\_transform(input\_data)  
  
 op = model.predict(scaled\_input\_data)  
 logging.debug("Model output: %s", str(op))  
  
 return render\_template('input.html', Output=str(op))  
  
 except Exception as e:  
 logging.error("Error during prediction: %s", str(e))  
 return render\_template('input.html', Output="Error: " + str(e))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)

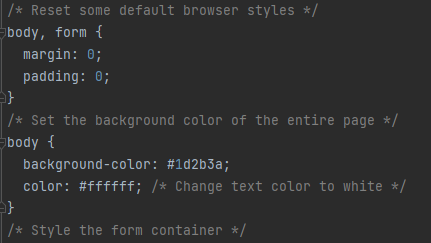
**HTML code:**

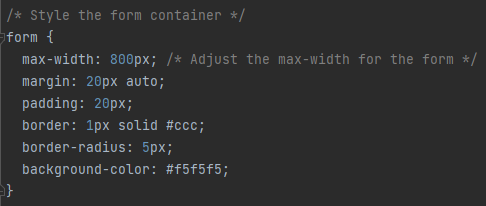


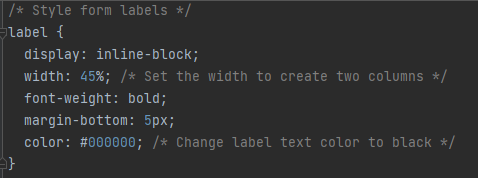


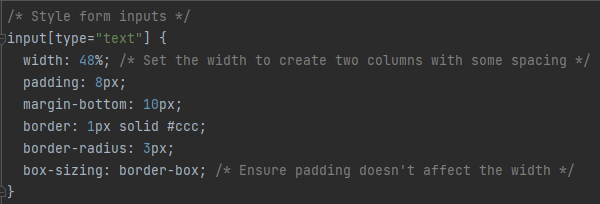


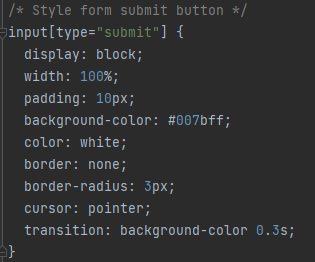
**CSS code:**

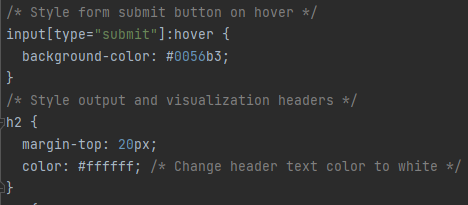


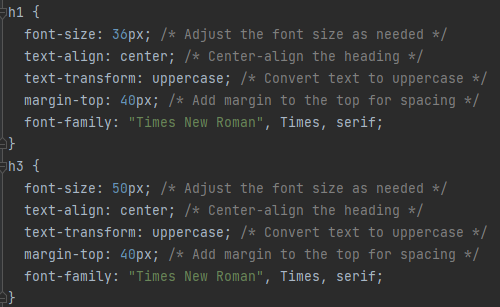


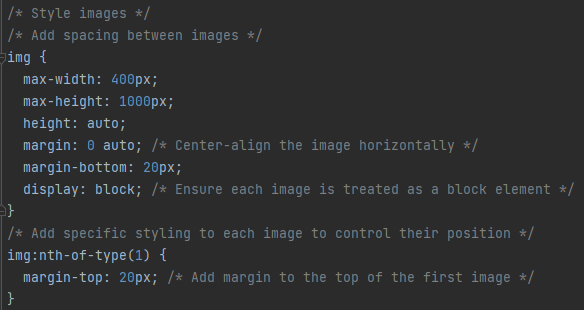


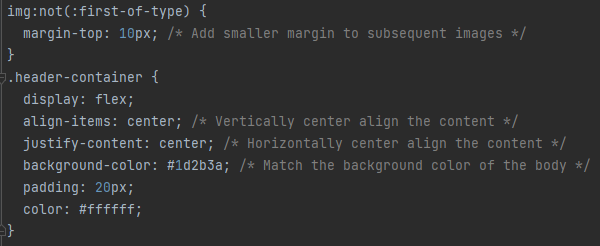


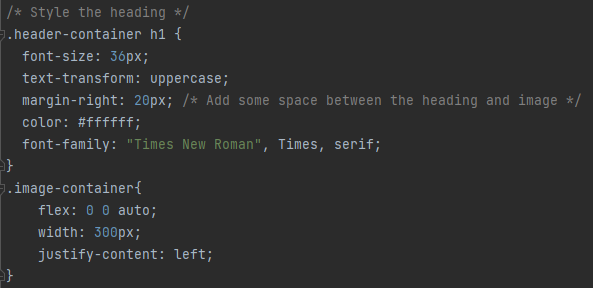


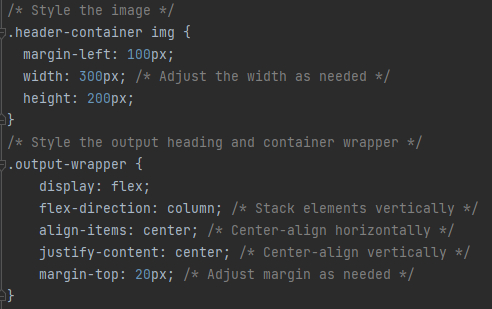


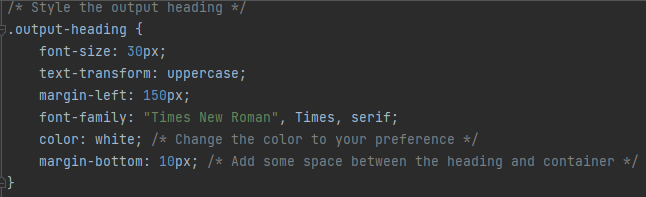


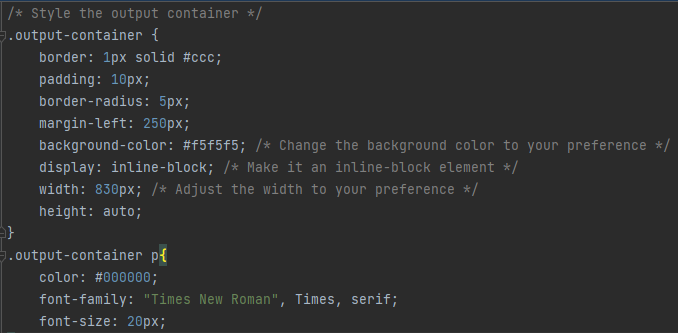




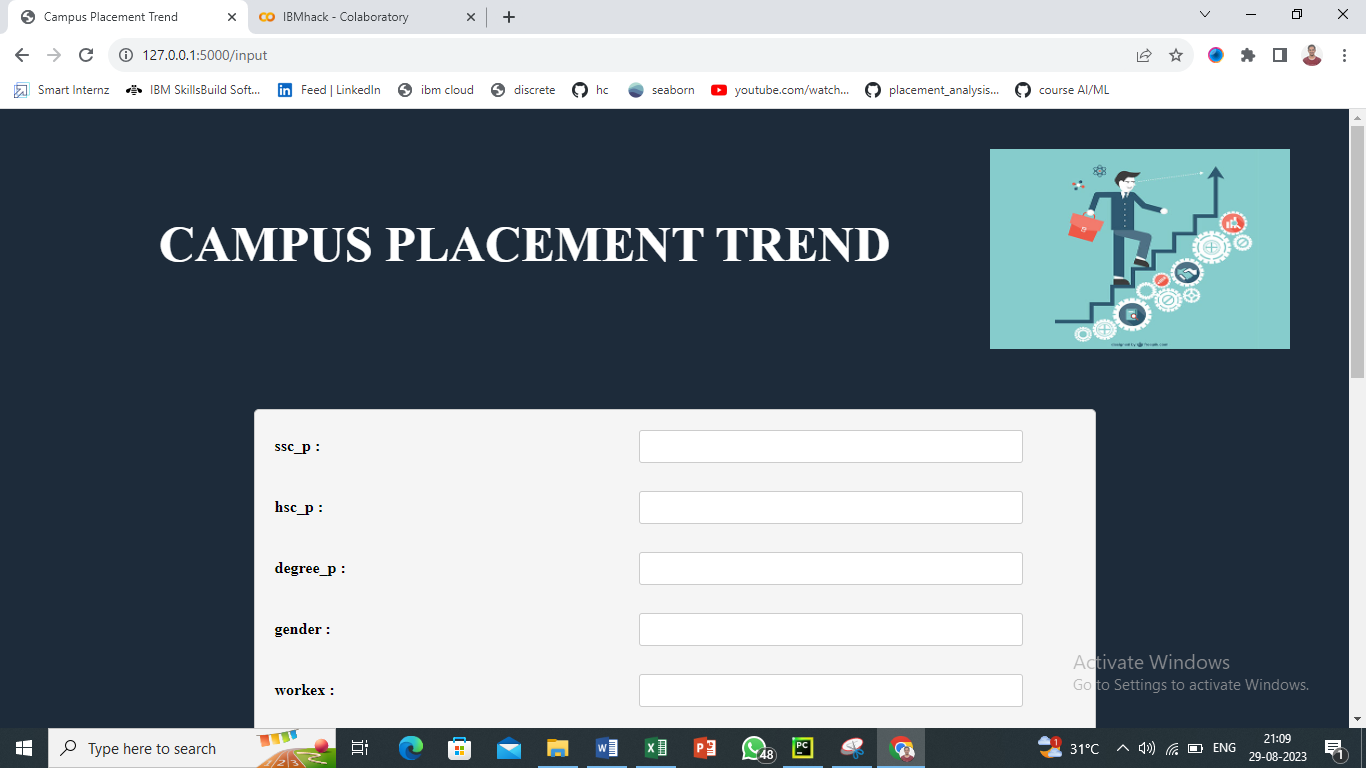


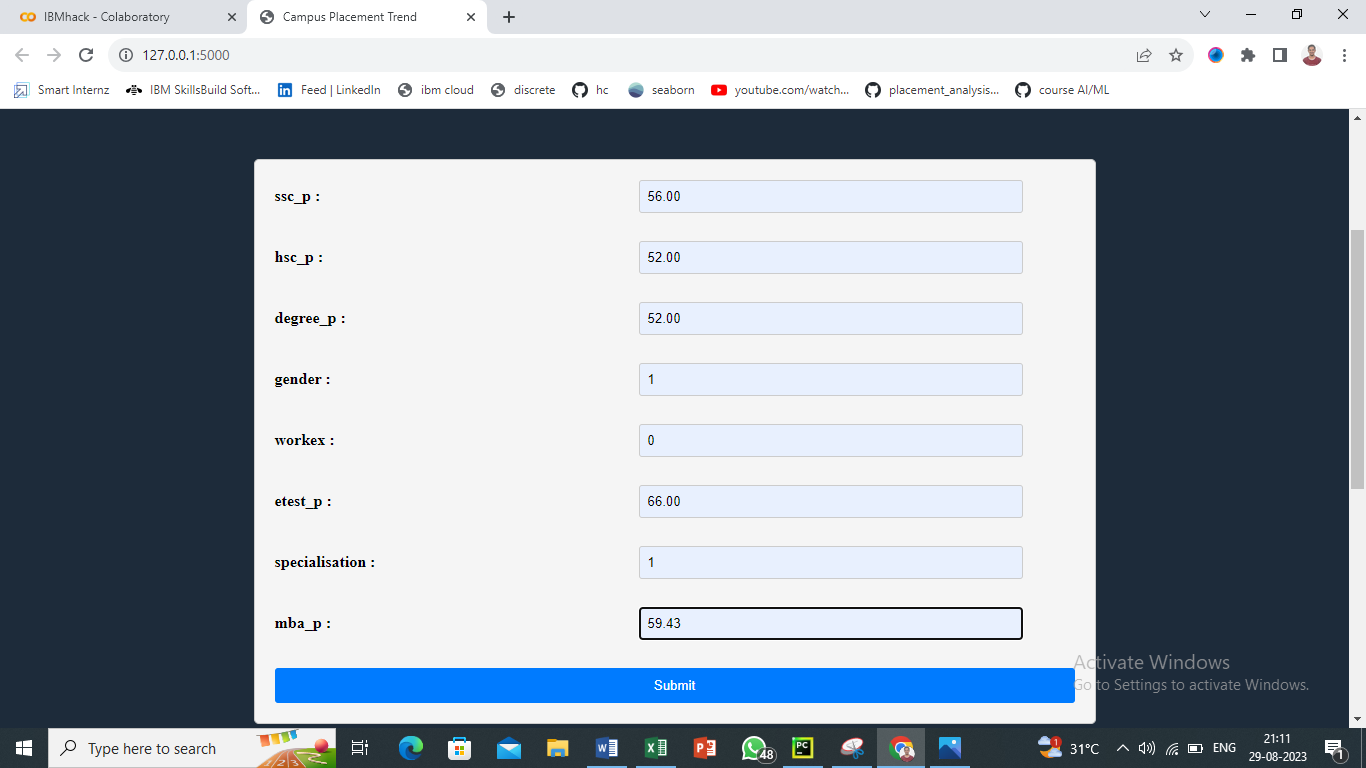


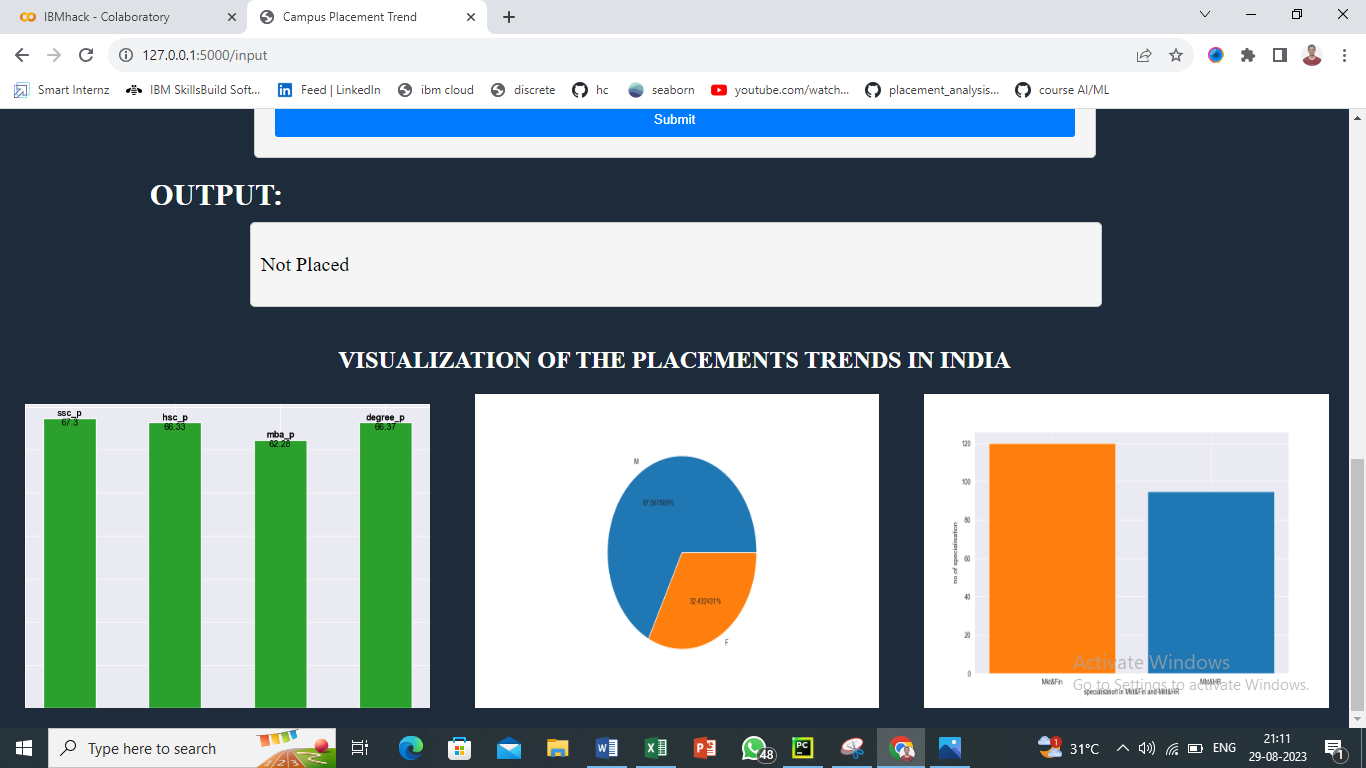




**Website:**

****

****

****

**References**

* Kaggle dataset: [https://www.kaggle.com/datasets/tejashvi14/engineering-placements-prediction](https://www.kaggle.com/datasets/benroshan/factors-affecting-campus-placement)
* <https://www.ibm.com/academic/home>
* <https://www.ibm.com/academic/topic/data-science>